CSE221 Data Structures (Fall 2016)
Instructor: Prof. Won-Ki Jeong
Due date: Nov 20, 2016, 11:59 pm.

Assignment 4: Spell checker (100 pts)

In this assignment, you will implement hash map data structure to check the spelling of a given word (i.e., spell checker). Spell checker is a program that searches the input word in the dictionary and if the word is not a dictionary word then suggests similar dictionary words. For example, if your input word is `stori`, then it is not a dictionary word. So, your program should suggest dictionary words similar to `stori`, such as `store, stork, storm`, and `story`. In order to quickly check whether the input word is in the dictionary, you should use a hash map.

1. Hash Map data structure (40 pts)

Hash Map is a template class defined as below:

```
template <class KeyType, class ValType>
class MapElem
{
public:
     typedef KeyType ktype;
     typedef ValType vtype;

     KeyType key;
     ValType val;

     MapElem* link;
};

template <class HashMapElemType>
class HashMap
{
     public:
          HashMap();
          ~HashMap();
     private:
          ...
};
```

Hash Map class should provide the functions below:

`HashMap(unsigned int c)`
Constructor with initial hash table size of c.

`int size()`
Return the size of the key-value pair

`bool isEmpty()`
Return true if no key-value pair is in the map

`HashMapElem* find(const key k)`
If k is in the map, return the pair (k,v) as MapElem. If find is not successful, return NULL.

`void insert(const key k, const value v)`
Insert a pair (k, v) in the map. If k is already in the map, change its value with v.

`bool remove(const key k)`
If k is in the map, remove its pair (k,v) and return true. If k is not in the map, return false.

`void print()`

Print all key:value pair in decreasing order of value.

`unsigned int hashfunction(const KeyType k)`

Returns an integer hash value converted from a key k.

You need to build a hash map with a given word as a key. You are required to implement the static hashing with the *division hash function* coupled with *chaining* for overflow handling. For a given non-numerical key (i.e., string), you need to convert it to an integer number (hash code) using the algorithm discussed in the class.

2. Spell checker (50 pts)

The main idea of the spell checker is searching the input word in the dictionary quickly using a hash map. If the word is not listed in the dictionary (i.e., search in the hash map is unsuccessful), then you need to find some other words in

the dictionary similar to the input word to suggest for correction. There might be several different suggestion algorithms, but we will use a very simple algorithm as follows:

- The length of suggested word is same as that of input word
- Only one letter (character) should be different

For example, if the input word is `kat`, then your suggestion can be `hat`, `cat`, `kit`, etc. In order to find the suggestions, you can replace each letter with a~z and search in the dictionary if it exists. For example, you can search `aat`, `bat`, `cat`, `dat`, `eat`, `...`, `zat`, `kbt`, `kct`, `...`, `kzt`, `kaa`, `kab`, `...`, `kaz`.

In order to do this, first you need to load the dictionary file (dictionary.txt) and add each word to the hash map (parser code is included in `main.cpp`). Note that, for simplicity, we replace all the upper case letters to lower case letters.

In `main.cpp`, you need to implement the following function:

```
void spellcheck(std::string s)
```

If you execute `spellcheck(word)`, then it will search the input word in the dictionary and print out the messages for 3 different cases.

(1) If the input word is found

```
> hello
> hello is in the dictionary
```

(2) If the input word is not found and there are some suggestions

```
> hrll
> hrll is NOT in the dictionary
> hrll : hall, hell, hill, hull
```

(3) If the input word is not found and there is no suggestion

```
> unist
> unist is NOT in the dictionary
> unist : no suggestion
```

Make sure you follow all the punctuation, formatting, and space shown above (for example, there is no comma at the end, and there is a space after comma. Each line starts with "> ").

3. Compile and submit

You must log in uni06~10.unist.ac.kr for coding and submitting the assignment. You can compile the code using the included Makefile. You can simply `make` and then the code will be compiled. The output executable name is `assign_4`.

Once you are ready to submit the code, use the `dssubmit` script as follows:

```
> dssubmit assign4 assign4.zip
```

You need to submit source code and the report (10 pts)

Good luck and have fun!